SALT : Software for Adversarial Life Testing

by

Simon P. Wilson
The George Washington University, Washington, D.C 20052

GWU/IRRA/Serial TR-91/12
December 1991

Research Supported by

Contract N00014-85-K-202
Office of Naval Research

Grant DAAL03-87-K-0056
The Army Research Office

and

Grant AFOSR-89-0381
The Air Force Office of Scientific Research

SALT : Software for Adversarial Life Testing

by

Simon Wilson

The George Washington University, Washington, D.C. 20052

This report describes software that has been developed for calculations appearing in Lindley & Singpurwalla (1990). It will begin with a brief description of the paper and then outline the program and what it does.

The paper by Lindley & Singpurwalla concerns itself with adversarial life testing. In particular it considered the situation where a manufacturer must decide whether to offer a sample of his product for testing in order to convince a consumer of the reliability of that product. If testing is to be done then it considers how many of the product should be tested. This is a decision problem and so first a probability model and set of utilities need to be developed. Then a simple criterion for whether testing should begin is given, and how many products should be tested if testing is to occur; the latter is done by picking the number of products to be tested that maximize expected utility.

SALT is a program that calculates these expected utility values. These values are a function of many parameters, related to both the utility function and the probability model, which need to be specified before the program can be run. A list of the parameters needed, and a brief description of their meaning, is given below.

a_1, a_2, a_3 and p : these specify the consumer's utility. The consumer's utility of accepting a product with observed lifelength x is $a_1x^p - a_2$ (so $-a_2$ is the utility to the consumer of accepting an item that does not work). The utility of rejecting a product is a constant a_3. SALT automatically sets p to 1.

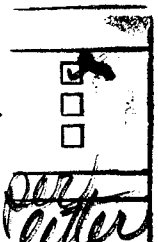
b_1, b_2, b_3 and q : these specify the manufacturer's utility. The manufacturer's utility is a mirror image of the consumer's ; the utility to the manufacturer of product acceptance with an observed lifelength x is $b_1x^p - b_2$, whilst the utility of rejection is b_3. Unlike p, SALT allows q to be specified by the user.

b_4 : the fixed cost to the manufacturer of putting an item on test.

b_5 : the fixed cost to the manufacturer of an item failing on test.

b_6 : the cost of running the test is assumed proportional to the total time on test. The constant of proportionality is b_6; so the cost of running the test for 1 unit of time is b_6.

α_1 and β_1 : the consumer is assumed to express his opinion about the mean lifelength of the product



that is being tested *before the test is done* by an inverse prior distribution, with scale parameter α_1 and shape parameter β_1.

α_2 and β_2 : the manufacturer also expresses his (possibly different) prior opinion about the mean lifelength of the product on test by an inverse gamma distribution with scale α_2 and shape β_2.

N : the size of the batch from which products may be drawn for testing; obviously the most products that can be tested is N.

n : this is a SALT parameter. SALT will calculate utilities for testing upto n items, where $n \leq N$. The user will find that calculations for n>15 become very time consuming.

SALT is designed to be user-friendly, and is menu-driven. The first task for the user is to specify a valid set of the parameters that have just been described; this is done by selecting the input parameters option. For each parameter a brief description of what it represents will be given, and the user is also free to select the default value which has been derived from MIL STD 781C, a standard testing plan used by the US Government. To prevent overflow it is recommended that the values of the parameters α_1, β_1, α_2, β_2 and q be not too large, and in particular that α_2, β_2 and q are less than about 70. Elicitation of the prior parameters α_1, β_1, α_2 and β_2 may not only be done by direct specification, but by through the prior elicitation option. This option relates them to the standard MIL STD 781C testing plan, which requires specification of a minimum acceptable mean time between failure, specified mean time between failure, manufacturer's and consumer's risks and so on; using this information the program fits an appropriate inverse gamma prior; see Lindley and Singpurwalla for more details.

Once the parameters have been entered, the user can then proceed with the solving of the decision problem or return to alter parameter values. Whilst the computer is finding utility values, it displays the current status of its calculations. To calculate utility values up to about n=12 takes very little time, but calculations for n=13 and higher become progressively slower. Beyond n=15 they can be prohibitively slow. Once calculations have been completed then the user can display the results in a table or as a graph, or alternatively send the output to a printer.

Reference

Lindley, D.V. and Singpurwalla, N.D. (1990) Adversarial Life Testing. Technical Report GWU/IRRA/Serial TR-90/5, The Institute for Reliability and Risk Analysis, The George Washington University, Washington, D.C.